## REMARKS

Reconsideration of the application is respectfully requested in view of the foregoing amendments and following remarks.

### Patentability Over CORBA in view of Steinman and Hamilton

The Office has asserted a rejection of claims 13, and 18-21, under 35 U.S.C. § 103(a) over "The Common Object Request Broker: Architecture and Specification, CORBA," Revision 2.0, July 1995 ("CORBA") in view of Steinman, J., "Incremented State Saving in SPEEDS Using C++,"" Proceedings of the 1993 Winter Simulation Conference ("Steinman") and Hamilton et al., "Subcontract: A Flexible Base for Distributed Programming," Association for Computing Machinery, 1993 ("Hamilton"). Applicants respectfully traverse.

**Claim 13**

Claim 13 is generally directed to a method of enhancing scalability of server applications comprising "discarding the processing state of the component responsive to the component indicating completion of the work before receiving any indication from the client that the component's work is complete."

Specifically, claim 13 recites,

> 13. (Once Amended) In a computer, a method of encapsulating state of processing work for a client by a server application in a component with improved scalability, comprising:
>
> encapsulating function code and a processing state for the work in a component;
>
> providing a reference through an operating service for a client to call the function code of the component to initiate processing of the work by the component;
>
> receiving an indication from the component that the work by the component is complete; and
>
> discarding the processing state of the component responsive to the component indicating completion of the work before receiving any indication from the client that the component's work is complete. (Emphasis Added).

Thus, the component can indicate discarding its own state upon completion of the work without any indication from the client that the component's work is complete.

The Examiner asserts that the claimed arrangement is obvious in light of a CORBA-Steinman-Hamilton combination. Applicants disagree. A CORBA-Hamilton-Steinman combination does not teach or suggest "discarding the processing state of the component responsive to the component indicating completion of the work before receiving any indication from the client that the component's work is complete."

First, the Examiner admits that CORBA fails to teach or suggest the recited arrangement. *See* Office Action, mailed June 21, 2002, page 3, ¶ 2. Second, the Examiner does not allege that Steinman teaches or suggests the recited arrangement. Since the references when combined "must teach or suggest all the claim limitations," Hamilton must teach or suggest the recited arrangement or the Examiner's proposed combination fails. But since Hamilton also fails to teach or suggest "discarding the processing state of the component responsive to the component indicating completion of the work before receiving any indication from the client that the component's work is complete," the Examiner has again failed to establish a prima facie case of obviousness.

For example, the Examiner asserts that the following Hamilton passage, discloses the recited arrangement. *See* Office Action, mailed June 21, 2002, page 3, ¶ 4.

> **5.2.3 Revoking an object**
> Occasionally, a server will decide that it wishes to discard a piece of state, event though there are clients who currently have objects pointing at that state. This particularly is important for operating system services which may wish to reclaim resources without waiting for all their clients to consent. Thus, typical server-side subcontracts provide a way for the server application to revoke an outstanding object. For example, this can be implemented by revoking any underlying kernel doors, which will effectively prevent further incoming kernel calls. (Hamilton, page 74, ¶ 5.2.3.)

In summary, Hamilton describes a server "that will decide ... to discard a piece of state, even though there are clients who currently have objects pointing to that state." Id. Thus, Hamilton describes, "a way for the server application to revoke an outstanding object." Id. Thus, the server is discarding state of an object held by a client. Note however, the object in Hamiltion is not making any indication to the server to discard its own state. Thus, Hamilton fails to teach or suggest "discarding the processing state of the component responsive to the component indicating completion of the work before receiving any indication from the client that the component's work is complete."

Hamilton fails to address the limitations recited in claim 13. Claim 13 recites (1) discarding the processing state of the component (2) responsive to (3) the component indicating completion of the work (4) before receiving any indication from the client that the component's work is complete. Hamilton does not teach or suggest this. The Examiner continues to ignore the fact that in the cited art, (1) a component is not indicating discarding its own state, and (2) a component is not requesting the destruction of its own state.

Again, the server in Hamilton is discarding state of an object held by a client. Note however, that the object held by the client is not requesting its own destruction. The Examiner continues to ignore that the claim language requires a component requesting its own states destruction.

For example, Hamilton states that some "objects happen to keep all their interesting state at a remote site, so that their local state merely consists of a handle to this remote site." See Hamilton, page 71, and Figure 2. Thus, Hamilton discusses a distributed object which maintains at least a portion of its state at a remote site. However, notice that nowhere does Hamilton state that this distributed object requests destruction of its own state. Rather, Hamilton discusses that a "server will decide that it wishes to discard" the state of an object. The object in Hamilton never indicates discarding its own state.

Therefore, Hamilton fails to teach or suggest "discarding the processing state of the component responsive to the component indicating completion of the work before receiving any indication from the client that the component's work is complete."

For at least this reason claim 13 should be allowed. Such action is respectfully requested.

**Claim 14**

Claim 14 depends from claim 13. Since claim 14 depends from claim 13, it should be allowed for at least the reasons stated for claim 13. In view of the foregoing discussion claim 13, the merits of the separate patentability of dependent claim 14 is not belabored at this time. Claim 14 should be allowed. Such action is respectfully requested.

**Claim 18**

Claim 18 is generally directed to a system service comprising "receiving an indication from the application component that processing by the application component of the work is complete ... and ... destroying the processing state of the application component without action from the client program." Specifically, claim 18 recites,

> 18. (Once Amended) In a computer, a system service for providing an execution environment for scalable application components, comprising:
>
> code responsive to a request from a client program to create an application component for returning to the client program a reference through the system service to the application component;
>
> code responsive to a call from the client program using the reference for initiating processing of work by the application component, the application component producing a processing state during processing the work;
>
> code for receiving an indication from the application component that processing by the application component of the work is complete; and
>
> code for destroying the processing state of the application component without action from the client program. (Emphasis Added).

The Examiner asserts that the claimed arrangement is obvious in light of a CORBA-Steinman-Hamilton combination. Applicants disagree. A CORBA-Hamilton-Steinman combination does not teach or suggest "receiving an indication from the application component that processing by the application component of the work is complete ... and ... destroying the processing state of the application component without action from the client program."

First, the Examiner admits that CORBA fails to teach or suggest the recited arrangement. *See* Office Action, mailed June 21, 2002, page 3, ¶ 2. Second, the Examiner does not allege that Steinman teaches or suggests the recited arrangement. Since the references when combined "must teach or suggest all the claim limitations," Hamilton must teach or suggest the recited arrangement or the Examiner's proposed combination fails. But since Hamilton also fails to teach or suggest "receiving an indication from the application component that processing by the application component of the work is complete ... and ... destroying the processing state of the application component without action from the client program," the Examiner has again failed to establish a prima facie case of obviousness.

For example, the Examiner asserts that the following Hamilton passage, discloses the recited arrangement. *See* Office Action, mailed June 21, 2002, page 3, ¶ 4.

> **5.2.3 Revoking an object**
> Occasionally, a server will decide that it wishes to discard a piece of state, event though there are clients who currently have objects pointing at that state. This particularly is important for operating system services which may wish to reclaim resources without waiting

> for all their clients to consent. Thus, typical server-side subcontracts provide a way for the server application to revoke an outstanding object. For example, this can be implemented by revoking any underlying kernel doors, which will effectively prevent further incoming kernel calls. (Hamilton, page 74, ¶ 5.2.3.)

In summary, Hamilton describes a server "that will decide ... to discard a piece of state, even though there are clients who currently have objects pointing to that state." Id. Thus, Hamilton describes, "a way for the server application to revoke an outstanding object." Id. Thus, the server is discarding state of an object held by a client. Note however, the object in Hamiltion is not making any indication to the server to discard its own state. Since, Hamilton fails to teach or suggest "receiving an indication from the application component that processing by the application component of the work is complete ... and ... destroying the processing state of the application component without action from the client program," the Examiner has again failed to establish a prima facie case of obviousness.

Hamilton fails to address the limitations recited in claim 18. Claim 18 recites (1) receiving an indication (2) from the application component (3) that processing by the application component is complete, and (4) destroying the processing state of the application component, (5) without action from the client program. Hamilton does not teach or suggest this. The Examiner continues to ignore the fact that in the cited art, (1) no indication is received from the application component, (2) no indication is received from the application component that processing is complete, (3) and no application component is indicating its own states destruction when processing is complete.

Again, the server in Hamilton is discarding state of an object held by a client. Note however, that the object held by the client is not requesting its own state's destruction. The Examiner continues to ignore that the claim language requires a component requesting its own states destruction. The object held by the client in Hamilton is not requesting anything, and no indication is received from the object.

For example, Hamilton states that some "objects happen to keep all their interesting state at a remote site, so that their local state merely consists of a handle to this remote site." See Hamilton, page 71, column 2, describing Hamilton Figure 2. Thus, Hamilton discusses a distributed object which maintains at least a portion of its state at a remote site. However, notice that nowhere does Hamilton state that this distributed object requests destruction of its own state. Rather, Hamilton discusses that a "server will decide that it wishes to discard" the state of an object. See Hamilton,

page 74, ¶ 5.2.3. The object in Hamilton never decides or requests to discard or destroy its own state. Therefore, Hamilton fails to teach or suggest "receiving an indication from the application component that processing by the application component of the work is complete ... and ... destroying the processing state of the application component without action from the client program"

For at least this reason claim 18 should be allowed. Such action is respectfully requested.

**Claim 19**

Claim 19 depends from claim 18. Since claim 19 depends from claim 18, it should be allowed for at least the reasons stated for claim 18. In view of the foregoing discussion claim 18, the merits of the separate patentability of dependent claim 19 not belabored at this time. Claims 19 should be allowed. Such action is respectfully requested.

**Claim 20**

Claim 20 depends from claim 18. Since claim 20 depend from claim 18, it should be allowed for at least the reasons stated for claim 18. In view of the foregoing discussion claim 18, the merits of the separate patentability of dependent claim 20 is not belabored at this time. Claim 20 should be allowed. Such action is respectfully requested.

**Claim 21**

Claim 21 is generally directed to a method of enhancing scalability of server applications comprising "destroying the state by the operating service in response to an indication from the application component without action by the client."

Specifically, claim 21 recites,

> 21. (Once amended) In a computer having a main memory, a method of enhancing scalability of server applications, comprising:
> executing an application component under control of an operating service, the application component having a state and function code for performing work responsive to method invocations from a client;
> maintaining the state in the main memory between the method invocations of the function code by the client in the absence of an indication from the application component that the work is complete; and
> destroying the state by the operating service in response to an indication from the application component without action by the client, such that the destroyed state is not persistent. (Emphasis Added).

The Examiner asserts that the claimed arrangement is obvious in light of a CORBA-Steinman-Hamilton combination. Applicants disagree. A CORBA-Hamilton-Steinman combination

does not teach or suggest "destroying the state by the operating service in response to an indication from the application component without action by the client."

First, the Examiner admits that CORBA fails to teach or suggest the recited arrangement. *See* Office Action, mailed June 21, 2002, page 3, ¶ 2. Second, the Examiner does not allege that Steinman teaches or suggests the recited arrangement. Since the references when combined "must teach or suggest all the claim limitations," Hamilton must teach or suggest the recited arrangement or the Examiner's proposed combination fails. But since Hamilton also fails to teach or suggest "destroying the state by the operating service in response to an indication from the application component without action by the client," the Examiner has again failed to establish a prima facie case of obviousness.

For example, the Examiner asserts that the following Hamilton passage, discloses the recited arrangement. *See* Office Action, mailed June 21, 2002, page 3, ¶ 4.

> **5.2.3 Revoking an object**
> Occasionally, a server will decide that it wishes to discard a piece of state, event though there are clients who currently have objects pointing at that state. This particularly is important for operating system services which may wish to reclaim resources without waiting for all their clients to consent. Thus, typical server-side subcontracts provide a way for the server application to revoke an outstanding object. For example, this can be implemented by revoking any underlying kernel doors, which will effectively prevent further incoming kernel calls. (Hamilton, page 74, ¶ 5.2.3.)

In summary, Hamilton describes a server "that will decide ... to discard a piece of state, even though there are clients who currently have objects pointing to that state." Id. Thus, Hamilton describes, "a way for the server application to revoke an outstanding object." Id. Thus, the server is discarding state of an object held by a client. Note however, the object in Hamiltion is not making any indication to the server to discard its own state. Since, Hamilton fails to teach or suggest "destroying the state by the operating service in response to an indication from the application component without action by the client, such that the destroyed state is not persistent."

Hamilton fails to address the limitations recited in claim 21. Claim 21 recites (1) destroying the state [of the application component] (2) by the operating service (3) in response (4) to an indication (5) from the application component (6) without action by the client. Hamilton does not

teach or suggest this. The Examiner continues to ignore the fact that in the cited art, a component is not requesting the destruction of its own state.

Again, the server in Hamilton is discarding state of an object held by a client. Note however, that the object held by the client is not requesting its own destruction. The Examiner continues to ignore that the claim language requires a component requesting its own states destruction.

For example, Hamilton states that some "objects happen to keep all their interesting state at a remote site, so that their local state merely consists of a handle to this remote site." See Hamilton, page 71, and Figure 2. Thus, Hamilton discusses a distributed object which maintains at least a portion of its state at a remote site. However, notice that nowhere does Hamilton state that this distributed object requests destruction of its own state. Rather, Hamilton discusses that a "server will decide that it wishes to discard" the state of an object. The object never decides to discard its own state.

Therefore, Hamilton fails to teach or suggest "destroying the state by the operating service in response to an indication from the application component without action by the client, such that the destroyed state is not persistent."

For at least this reason claim 21 should be allowed. Such action is respectfully requested.

## CONCLUSION

The claims in their present form should now be allowable. Such action is respectfully requested.

Respectfully submitted,

KLARQUIST SPARKMAN, LLP

By _____
Stephen A. Wight
Registration No. 37,759

One World Trade Center, Suite 1600
121 S.W. Salmon Street
Portland, Oregon 97204
Telephone: (503) 226-7391
Facsimile: (503) 228-9446
(80683.1USORG)

**Marked-up Version of Amended Claims**
**Pursuant to 37 C.F.R. § 1.121(c)**

6. (Canceled) ~~The computer operating environment of claim 5 wherein the indication is a call from the client to commit or abort a transaction encompassing the work.~~

18. (Once Amended) In a computer, a system service for providing an execution environment for scalable application components, comprising:

code responsive to a request from a client program to create an application component for returning to the client program a reference through the system service to the application component;

code responsive to a call from the client program using the reference for initiating processing of work by the application component, the application component producing a processing state during processing the work;

code for receiving an indication from the application component that processing by the application component of the work is complete; and

code for destroying the processing state of the application **component** ~~program~~ without action from the client program.